UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Centre de la Imatge i la Tecnologia Multimèdia

Prototipo de videojuego de Terror y Puzzle en Realidad Virtual

Trabajo Final de Grado

Grado en Diseño y desarrollo de Videojuegos

Apellidos: Iglesias Gavarró Nombre: Asier

Plan: 2014

Director: Torelló Oliver, Josep

Índice

Resumen

Este documento corresponde a la descripción del trabajo para el desarrollo del prototipo del videojuego de Terror y Puzzle en Realidad Virtual “The Evil Sage”. El objetivo de este trabajo es que el seguimiento de la creación del prototipo sirva como base para aquellos que tengan la intención de crear un videojuego parecido. Para evitar que cometan los errores que se hayan producido durante el desarrollo y que cuenten con la experiencia previa de otra persona.

El resultado obtenido es un prototipo jugable con las gafas de Realidad Virtual HTC Vive, y en el documento se explica todo el proceso del desarrollo al detalle, incluyendo los problemas encontrados durante la ejecución y las conclusiones sacadas del proceso.

Palabras clave

Videojuego, Realidad Virtual, Terror, Puzzle, Unity, C#, Prototipo, Diseño, Programación.

Enlaces

URL donde descargar el prototipo:

https://github.com/asierigle/TheEvilSage/releases/tag/v0.1

URL Gameplay del prototipo:

https://youtu.be/V8JBeYsYJQs

Índice de tablas

Índice de figuras

Glosario

Asset: Cada uno de los elementos que componen el juego (animaciones, modelos, IA, sonidos, etc).

Bool: Tipo de dato informático que guarda dos posibles datos, o 1 o 0, también conocidos como verdadero o falso.

Collider: Componente que define la forma de un objeto para los propósitos de colisiones físicas.

Far plane: Plano del objeto de la cámara que corresponde a la posición máxima que se llega a ver.

Layer de un objeto: La capa en la que se encuentra un objeto.

Milestones: Un evento significativo que ocurre durante el proyecto, que generalmente coincide con la terminación de un entregable principal del proyecto.

Plugin: Es aquella aplicación que, en un programa informático, añade una funcionalidad adicional o una nueva característica al software.

Screamer: Imagen, animación o sonido con el objetivo de asustar a alguien.
Script: Es un programa, usualmente simple, que por lo regular se almacena en un archivo de texto plano.

Survival horror: Es un subgénero perteneciente al género de videojuegos conocido como aventuras de acción, caracterizado por combinar elementos de aventura gráfica con elementos de suspenso y acción, dentro de una atmósfera o ambientación de terror psicológico.

Tester: Persona que prueba algo para ver si se cumple con los resultados deseados.

VR: Abreviación de las palabras en inglés Virtual Reality, que significa Realidad Virtual.

1. Introducción

1.1 Motivación

Me ha gustado jugar a videojuegos desde que tengo uso de razón. El problema era que mis padres tenían una opinión muy mala sobre los videojuegos. No me dejaban tener ninguna consola y apenas podía jugar en el ordenador, por lo que aprovechaba siempre que podía para jugar con amigos fuera de casa. Lo que causó que desarrollara un interés aún mayor para jugar, que años más tarde desencadenó en querer crear videojuegos como forma de vida.

Entre las memorias que más aprecio, por ser una de las más felices que he vivido, se desarrolla en torno a un videojuego llamado *Slender: The Eight Pages*[1], un videojuego en primera persona de survival horror que salió en 2012. En esta memoria estamos catorce amigos y amigas en el salón de un amigo, mientras uno juega los demás estamos mirando en la televisión, gritando en los momentos que dan más miedo y riendo a continuación.

Una de las principales motivaciones que tengo para hacer este trabajo está relacionada con la memoria que acabo de contar. Me encantaría poder hacer que diferentes personas al jugar a este videojuego, si están con seres queridos alrededor, puedan sentir lo que sentí ese día hace tantos años.

Otra de las motivaciones es retarme a mí mismo. Quiero ver si soy capaz de crear un prototipo de un juego y unas pautas a seguir para poder llegar a objetivos similares al que quiero llegar.

La última motivación que tengo consiste en aprender a hacer un juego en realidad virtual. Desde que supe sobre la realidad virtual, todo sobre ella me ha fascinado y he querido hacer algo con ella, ya sea un trabajo de la universidad o un proyecto propio.

Hace un par de años se me ocurrió una idea muy simple que podría implementar haciendo un juego de miedo, por lo que decidí que este trabajo de final de grado fuera en torno a este tema, el miedo en los videojuegos.

1.2 Formulación del problema

La Realidad Virtual es una tecnología muy nueva y no hay una manera de desarrollar videojuegos de Terror para la Realidad Virtual predefinida. Si que hay diferentes tutoriales para crear pequeñas escenas con un uso básico del dispositivo VR (en mi caso me centraré en el HTC VIVE) y *discusiones en la que se dan consejos entre desarrolladores*[2]. Por lo que si alguien quiere hacer un juego en VR no hay una manera de saber exactamente qué hacer.

[1] Trailer del juego Slender: The Eight Pages: https://www.youtube.com/watch?v=QnvjNcV8kX8 consultado el 04/03/2019

[2] Página web de discusion entre desarrolladores: https://www.reddit.com/r/Vive/comments/506epe/learning_to_develop_vr_games_for_the_htc_vive/ consultado el 09/03/2019

1.3 Objetivos generales del TFG

El objetivo de este trabajo consiste en crear un prototipo de un videojuego de terror con características de puzzle y que el seguimiento del trabajo sirva de referencia para los que quieran hacer un juego similar en un futuro. El juego debe ser jugable en las HTC VIVE, unas de las gafas de Realidad Virtual más potentes y famosas del mercado, es decir utilizar el área de rastreo[3] que proveen los rastreadores de las gafas y también usar los mandos como si fueran las manos del jugador.

1.4 Objetivos específicos del TFG

El objetivo no es solo hacer un prototipo de videojuego de terror, sino que debe contar con algunas características:

- Que sea totalmente inmersivo: Cuando se juegue debe de generar la sensación de que se está en otro mundo.
- Que sea divertido y rejugable: Cuando se juegue durante diferentes periodos de tiempo que el juego sea aleatorio y que no aburra al jugador.
- Que cree un sentimiento de ansiedad al intentar superar los obstáculos presentados por el juego.
- Deberán haber 5 habitaciones diferentes. En cada una tienes que resolver un enigma y recoger un objeto (el cual debe ser la respuesta al enigma realizado) para pasar a la siguiente.
- 10 enigmas diferentes en cada habitación que se eligen de manera aleatoria al empezar a jugar.

Otro objetivo es aprender a utilizar la librería oficial hecha por Valve, *SteamVR SDK*[4].

Para hacer el trabajo colaboraré con mi compañero Josep Atencia que hace un análisis y construcción del Terror en los videojuegos, y aplicaré lo encontrado a mi proyecto.

1.5 Alcance del proyecto

El proyecto contará con una investigación previa al desarrollo del videojuego, un documento de diseño y el prototipo del juego mencionado en los objetivos generales y específicos.

El proyecto va dirigido a todo aquel que tenga interés por la realidad virtual y los videojuegos de terror y sobre todo a aquellos que tengan gafas de Realidad Virtual para poder jugar al resultado del proyecto.

Por último, los beneficiarios del resultado de este trabajo serán diferentes desarrolladores. Aquellos que quieran hacer juegos de Realidad Virtual y continuar unas pautas creadas anteriormente.

[3] Definido en el punto 2.1.1 apartado Área de rastreo
[4] Página web en la que se encuentra el SteamVR SDK: https://assetstore.unity.com/packages/tools/integration/steamvr-plugin-32647 consultado el 14/03/2019

2. Estado del arte

2.1 Realidad Virtual

Lo primero que necesitamos saber, es qué es la Realidad Virtual. Según la RAE su definición es la siguiente: *Representación de escenas o imágenes de objetos producida por un sistema informático, que da la sensación de su existencia real.*[5]

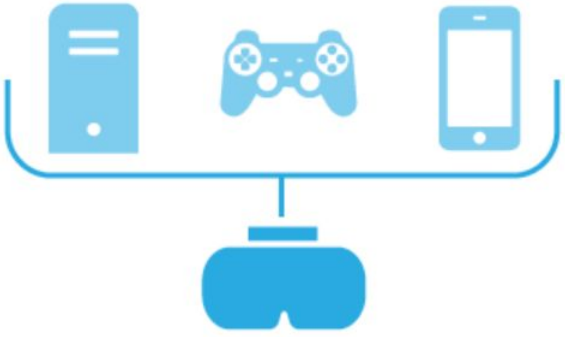
Lo que hoy en día se entiende como Realidad Virtual consiste en la inmersión sensorial en un nuevo mundo, basado en entornos reales o no, que ha sido generado de forma artificial, y que podemos percibir gracias a unas gafas de realidad virtual y sus accesorios (mandos, auriculares, sensores de posición...). Con esta tecnología se pueden crear mundos ficticios e interactuar con ellos de diversas formas.

Aunque aún no seamos demasiado conscientes, la Realidad Virtual es uno de los cambios tecnológicos más importantes de los últimos tiempos[6], ya que tiene la posibilidad de desarrollarse en muchos campos, como la medicina, educación, entretenimiento...

2.1.1 Principales características de gafas de Realidad Virtual

Hardware necesario

La mayoría de dispositivos de Realidad Virtual no funcionan de manera autónoma, por lo que es necesario disponer de otros dispositivos a los que conectarse para que las gafas de Realidad Virtual puedan ser funcionales. En el mercado se encuentran diferentes gafas que funcionan conectándose a diferentes hardwares.



F 2.1 - Dispositivos a los que se conectan diferentes gafas VR

En primer lugar podemos encontrarnos gafas VR que necesitan conectarse a un ordenador, el cual debe ser un ordenador con una potencia mínima para poder llevar a cabo los cálculos necesarios para la Realidad Virtual. En este segmento encontraremos los dispositivos VR con las mejores características del mercado: Oculus Rift, HTC VIVE o StarVR.

[5] Página web: https://dle.rae.es/srv/fetch?id=VH7cofQ consultado el 05/03/2019

[6] Página web: http://mundo-virtual.com/que-es-la-realidad-virtual/ consultado el 05/03/2019

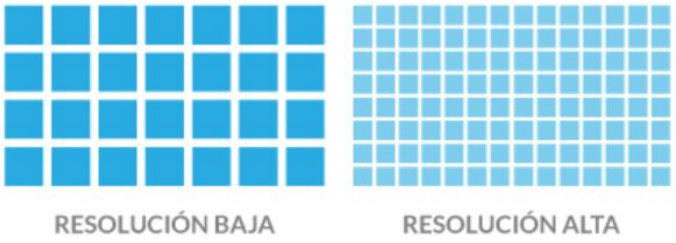
En segundo lugar nos encontraremos las gafas de Realidad Virtual que funcionan conectándose a un Smartphone, no solo usando su CPU sino que también utilizan su pantalla. Gracias a unas lentes bifocales en las gafas se produce un efecto de visión estereoscópica. En este grupo podemos encontrarnos Samsung Gear VR o Google Cardboard entre otros (ya que hay muchas empresas que han creado gafas de cartón con las lentes mencionadas anteriormente).

Y por último están las gafas VR destinadas a un uso lúdico y optimizadas para videojuegos, en este caso las gafas se conectan a una consola y se sirven de la CPU de ésta para su funcionamiento, como las PlayStation VR.

Resolución

La resolución de la imagen de la pantalla es un aspecto muy importante ya que el componente visual es la mayor parte de las gafas VR. Puesto que las pantallas se encuentran muy cerca de los ojos y el dispositivo necesita crear una sensación de Realidad Virtual, una mala calidad de imagen hace imposible la experiencia.

La resolución se mide en píxeles, que son pequeños cuadrados en la pantalla, y habitualmente se encuentra escrita como una multiplicación de dimensiones (alto x ancho). A mayor cantidad de píxeles mejor será la calidad de imagen, es decir cuantos más cuadraditos puedan iluminarse con diferentes colores mejor pueden representar en su conjunto cualquier imagen. En las gafas VR una mayor resolución hará que la imagen sea más nítida y no se aprecie una malla de cuadraditos, ya que como se ha mencionado anteriormente la pantalla está mucho más cerca que en los formatos tradicionales.



F 2.2 - Diferencia entre resolución baja y alta

Tasa de refresco

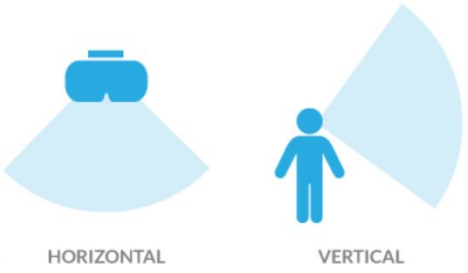
La tasa de refresco indica el grado de fluidez que tienen las imágenes de una pantalla. Al igual que la resolución esta característica es mucho más importante que en el resto de pantallas, ya que al realizar movimientos bruscos con la cabeza, si baja el refresco, la fluidez de la experiencia se perderá. Y no solo eso, sino que también puede provocar mareos y pérdidas de equilibrio que pueden afectar a la salud del usuario.

La tasa de refresco se mide en Herzios (Hz) por segundo, lo que nos indica las veces que se refresca la imagen por segundo, por lo que cuantas más veces se actualice la imagen el movimiento será más fluido. La cantidad de refresco óptima a la que se ha llegado en las gafas VR más famosas , Oculus Rift y HTC VIVE, es de 90Hz.

F 2.3 - Diferencia entre tasa de refresco baja y alta

Ángulo de visión

Cuanto más amplio sea el ángulo de visión más amplio será el campo de visión. Por lo que es necesario que las gafas VR cubran el ángulo de visión más amplio posible para que siga habiendo la sensación de estar en otro lugar que no sea la realidad. Esto se mide en grados, y los humanos tienen un ángulo de visión ligeramente superior a 164º en el eje horizontal y 130º en el eje vertical. Las gafas StarVR son las que mayor ángulo de visión presentan con 210º de visión horizontal.



F 2.4 - Referencia para los grados

Sensores

Para poder registrar los movimientos del jugador, saber la posición del jugador e interactuar con el dispositivo se necesitan una serie de sensores que pueden ir o no integrados en las propias gafas de realidad virtual. Entre los sensores que captan los movimientos de la cabeza se encuentran acelerómetros, giroscopios y magnetómetros que van integrados en las gafas. Por otro lado hay sensores de rastreo de posición que son externos, estos se colocan en la habitación y registran los movimientos que se efectúan dentro de un área específica. También existen otros sensores que mejoran la experiencia VR, como la cámara frontal de las gafas HTC VIVE.

Área de rastreo

El área de rastreo es la superficie en la cual se registran los movimientos de las gafas y los controladores que estas puedan tener. No todos los dispositivos de realidad virtual tienen esta función, pero se consigue una inmersión extra en el caso que la tengan. Las HTC VIVE son las gafas VR que mayor área de rastreo tienen, de 4,5 x 4,5 metros.

F 2.5 - Área de rastreo

2.1.2 Mecanismos de la Realidad Virtual

Estos son los 5 mecanismos más importantes de la Realidad Virtual:

Gráficos 3D

Gráficos tridimensionales que permiten tener una percepción real de lo que se ve a través de las gafas de realidad virtual.

Técnicas estereoscopía

Este tipo de técnicas es la que permite darle profundidad y realismo a las imágenes tridimensionales, es un efecto que se consigue con dos imágenes paralelas, “engañando” a la mente para que estas se superpongan y creen la sensación de profundidad.

Simulación del comportamiento

Los movimientos que va a seguir un personaje no están predefinidos sino que son improvisados y tienen múltiples variables, por lo que están en constante evolución.

Navegación

Para la movilidad a la hora de manejar un “avatar”, no solo se dispone un dispositivo como un mando, sino que la visión se fusiona con la aplicación sobre la que se está interactuando. Por lo tanto, los controles son mucho más intuitivos y es más fácil desenvolverse en un mundo virtual, ya que se desarrollan movimientos naturales.

Técnicas para una inmersión total

Las gafas de realidad virtual deben disponer de algo muy necesario, el aislamiento del mundo real. Para que la sensación de inmersión y realidad paralela que se vive sea lo más completa posible. Siendo el oído y la vista los sentidos que más estímulos reciben.

2.1.3 Dispositivos de Realidad Virtual

	HTC Vive Pro	HTC Vive	HTC Vive Focus	Oculus Rift	Oculus Go	Realidad mixta de Windows	PlayStation VR
Tipo de pantalla	AMOLED	OLED	AMOLED	OLED	-	LCD - AMOLED	OLED
Resolución	2880 x 1600	2160 x 1200	2880 x 1600	2160 x 1200	-	2880 x 1440	1920 x 1080
Tasa de refresco	90 Hz	90 Hz	75 Hz	90 Hz	-	90 Hz	90 Hz
Campo de visión	110°	110°	110°	110°	-	105°	100°
Área de movimiento	100 m2	20 m2	100 m2	6 m2	-	4 m2	-
Plataforma	SteamVR	SteamVR	Vive Wave	Oculus Home	-	Windows 10	PlayStation 4
Inalámbrico	Si (adaptador)	No	Si	No	Si	No	No
Precio	879€	599€	513€	449€	199$	desde 399€	399€

T 2.1 - Dispositivos de Realidad Virtual[7]

2.2 Videojuegos de terror en la Realidad Virtual

Para hacer un juego de terror de Realidad Virtual, lo más recomendable es hacer un estudio de los juegos de este género que más éxito hayan tenido en el mercado. A continuación se encuentra el estudio de tres videojuegos de Terror, los cuales han recibido muy buenas críticas y son considerados unos de los mejores juegos de Terror del mercado. Y por último, un cuarto juego, que más se acerca al objetivo al que se quiere llegar tras hacer este proyecto.

2.2.1 Arizona Sunshine

Ganador del mejor juego de Realidad Virtual 2016, es un survival zombie en el que puedes explorar diferentes zonas y seguir un modo historia. También se puede jugar con un amigo de manera cooperativa.

Mecánicas básicas:

- Movimiento por teletransportación hacia el sitio que apunta el mando.
- Recoger armas en diferentes lugares.
- Disparar a los zombies que tiene alrededor.

[7] Pagina web: https://omicrono.elespanol.com/2018/04/mejor-dispositivo-de-realidad-virtual/ consultado el 07/03/2019

F 2.6 - Captura del juego Arizona Sunshine[8]

2.2.2 Alien: Isolation (con el mod MotherVR)

Trata de una experiencia de horror y supervivencia que tiene lugar quince años después de los eventos de Alien. El objetivo del juego es intentar huir del Alien que te está cazando recolectando llaves y códigos que te permiten escapar de la estación espacial en la que se aparece.

Mecánicas básicas:

- Movimiento con controlador, no hay teletransportación.
- Esconderse en diferentes lugares para no ser cazado.
- Recoger objetos que ayudan a cumplir el objetivo del juego.

F 2.7 - Captura del juego Alien: Isolation[9]

2.2.3 Duck Season

Es un juego del estilo de *Five Nights at Freddy's*[10] en el que no solo tienes que estar quieto reaccionando a los acontecimientos que ocurren a tu alrededor, sino que también debes

[8] Video gameplay de Arizona Sunshine: https://www.youtube.com/watch?v=Nw5YzBqk01I consultado el 08/03/2019

[9] Video gameplay de Alien:Isolation con el mod MotherVR: https://www.youtube.com/watch?v=xjzJMaQqzGc consultado el 08/03/2019

[10] Página web del juego Five Nights at Freddy's: https://store.steampowered.com/app/319510/Five_Nights_at_Freddys/?l=spanish consultado el 08/03/2019

pasarte unos niveles del juego Duck Hunt adaptado a la Realidad Virtual y disparar en los momentos que necesites.

Mecánicas básicas:

- Recoger objetos y armas para disparar.
- Recargar el arma para poder seguir disparando.

F 2.8 - Captura del juego Duck Season[11]

2.2.4 Nevrosa: Prelude

Este es el juego que más se parece al objetivo que quiero llegar. Es un juego de terror con elementos de puzzle. Trata de una escape room en Realidad Virtual en la que debes resolver diferentes rompecabezas que se presentan uno tras otro, ya que estas encerrado en un laboratorio con una extraña criatura poco amistosa.

Mecánicas básicas:

- Resolver rompecabezas utilizando los mandos como si fueran tus manos.
- Movimiento por el área de rastreo.

F 2.9 - Captura del juego Nevrosa: Prelude[12]

[11] Video gameplay de Duck Season: https://www.youtube.com/watch?v=x-daxzVxrQI consultado el 08/03/2019

[12] Video gameplay de Nevrosa: Prelude: https://www.youtube.com/watch?v=Fq7jlZ6u7FE consultado el 08/03/2019

3. Gestión del proyecto

3.1 Procedimiento y Herramientas para el seguimiento del proyecto

3.1.1 Planificación

La mayoría de empresas en la industria de los videojuegos utilizan una metodología ágil llamada “Scrum”, que consiste en tener unos objetivos claros donde los sub-objetivos para llegar al objetivo principal se desarrollan de una manera ágil e iterativa. Estos sub-objetivos se pueden cambiar a lo largo del proyecto, ya que cada cortos periodos de tiempo, entre 10 y 14 días se hacen “Sprints” en la que se decide qué hacer con los objetivos que no se han llevado a cabo y cuáles serán los próximos objetivos.

En este proyecto se utilizará esta metodología con las siguientes milestones:

- Para el día 24/03/2019: Documento de diseño acabado. Siempre con posibles cambios, pero la primera versión deberá estar definida.
- Para el día 28/04/2019: Una versión jugable en la que estarán implementados la mayoría de aspectos del diseño.
- Para el día 01/06/2019: Una beta del videojuego ya casi finalizado.
- Para el día 20/06/2019: La versión final del prototipo.

Actualización 03/05/2019

EL documento de diseño me ha llevado más tiempo de lo esperado, y la versión jugable con los aspectos de diseño implementados aún está en proceso, por lo que haré cambios en la planificación de los milestones para poder llegar al objetivo deseado.

- Para el día 24/03/2019: Documento de diseño acabado. Siempre con posibles cambios, pero la primera versión deberá estar definida.
- Para el día 15/05/2019: Una versión jugable en la que estarán implementados la mayoría de aspectos del diseño.
- Para el día 15/06/2019: Una beta del prototipo ya casi finalizada.*
- Para el día 24/06/2019: La versión final del prototipo.

*En este punto hubo un error de escritura, ya que lo que haré es un prototipo y no un juego completo.

3.1.2 HacknPlan

Para el seguimiento del proyecto, se utilizará la herramienta HacknPlan. Ya que se puede llevar el seguimiento de proyectos de una manera excepcional, ver qué tareas están en progreso, cuales están finalizadas y las horas invertidas para cada una.

3.2 Herramientas de validación

Se organizarán diferentes sesiones de testeo para probar si el proyecto está yendo por buen camino. Estas sesiones estarán organizada 1 o 2 días después de cada milestone a partir de la del día 28/04/2019.

3.3. DAFO

	Positivos	Negativos
Origen Interno	**Fortalezas** ● Gracias a la metodología que utilizaré podré hacer cambios a medida que avanza el proyecto para conseguir el mejor resultado posible. ● Desarrollaré una idea en la que llevo pensando años, por lo que haré un esfuerzo extra.	**Debilidades** ● No sé qué especificaciones se necesitan para programar para la Realidad Virtual, por lo que puede que me encuentre con muchos problemas inesperados.
Origen Externo	**Oportunidades** ● Este proyecto se aprovecha del auge de la Realidad Virtual que cada vez se está haciendo más y más popular.	**Amenazas** ● Con la cantidad de juegos que salen a la venta a diario, será muy complicado conseguir que el juego tenga éxito en diferentes tiendas virtuales.

T 3.1 - DAFO

3.4. Riesgos y plan de contingencias

Los posibles riesgos identificados de este proyecto, y sus correspondientes soluciones son las siguientes, ordenadas de menor a mayor importancia:

Riesgo	Solución
El volumen de trabajo del modelado y texturizado de los objetos 3D sea mucho más grande de lo esperado.	Comprar assets online o subcontratar un artista.
No poder cumplir un objetivo por falta de conocimiento.	Pedir ayuda a alguien experto en el campo o a alguien que pueda saber algo sobre el tema que pueda proveer información útil.
Perder el proyecto entero por causas desconocidas.	Documentar todo el proceso para poder volver al mismo punto en el que se estaba.

T 3.2 - Plan de contingencias

3.5. Análisis inicial de costes

Los costes principalmente se componen de un salario personal y del equipamiento necesario para llevar a cabo el proyecto. También teniendo en cuenta que será necesario comprar assets y canciones de fuentes externas o subcontratar a alguien para que componga la música.

Type	Subject	Price	Type	Years of amortization	Total prize
Direct costs					
Personnel	Salary	2.600,00 €	Monthly		13.000,00 €
Equipment	HTC Vive	600,00 €	Amortization	2	125,00 €
	Computer	1.300,00 €	Amortization	3	180,56 €
	Screen	70,00 €	Amortization	4	7,29 €
	Keyboard	35,00 €	Amortization	4	3,65 €
	Mouse	20,00 €	Amortization	3	2,78 €
Supplies	Coworking	175,00 €	Monthly		875,00 €
	Assets	50,00 €	Monthly		250,00 €
	Musician	150,00 €	Monthly		750,00 €
	Unity	0,00 €	Monthly		0,00 €
	Github	0,00 €	Monthly		0,00 €
	Visual Studio	0,00 €	Monthly		0,00 €
Indirect costs					
Manteinance	Electricity	20,00 €	Monthly		100,00 €
	Water	12,00 €	Monthly		60,00 €
	Food	30,00 €	Monthly		150,00 €
Total					15.504,27 €

Duration of the Project in Months:
5

T 3.3 - Costes

Actualización 03/05/2019

Los costes siguen siendo exactamente los mismos, ya que estaba previsto que se produjera algún contratiempo y se tuvo en cuenta a la hora de presupuestar el proyecto.

4. Metodología

Para llevar a cabo este proyecto se usará una metodología de preproducción, producción y postproducción.

En la preproducción se encuentra la elaboración del GDD (Game Design Document, documento de diseño) y el aprendizaje del uso de las tecnologías y herramientas necesarias para la producción del proyecto.

En la fase de producción se llevará a cabo el juego en sí, estando aquí las tres últimas milestones mencionadas en el apartado 3.1.1 Planificación y los testeos de estas milestones.

Y por último en la fase de postproducción, una vez acabado el juego se hará un estudio en el que se verá si el juego está cumpliendo con los objetivos deseados, y si no es así, hacer los cambios pertinentes para cumplirlos.

Actualización 03/05/2019

Ahora mismo me encuentro en la fase de producción, el único cambio respecto a la metodología anterior es que la parte de producción va a ser más larga, pero el sistema de preproducción, producción y postproducción siguen siendo la metodología necesaria para el proyecto.

5. Desarrollo del proyecto

Para comenzar este proyecto lo primordial es tener una idea muy básica sobre cómo debe ser el videojuego. Después, para desarrollar esa idea, es importante invertir las horas necesarias en definir las bases del juego (historia, mecánicas básicas, gameplay...) y empezar a idear un documento de diseño como el que se encuentra en los anexos.

5.1 Tecnologías utilizadas en el proyecto

- Unity, versión *Unity 2018.3.6f1*[13]. Es el motor(programa) en el que se montará todas las partes completadas que forman parte de un videojuego. Esto permitirá crear un fichero ejecutable con el que se podrá jugar directamente con poco más de un click.

F 5.1 - Logo Unity

- El plugin para Unity de SteamVR versión *SteamVR 2.2.0*[14]. Es un conjunto de archivos que facilita la implementación de la realidad virtual en el videojuego.

F 5.2 - Logo SteamVR

- *Autodesk Maya*[15]. Es el programa en el que se hará el modelado y animación de todos los objetos y personajes 3D del juego.

F 5.3 - Logo Maya

[13] Página web: https://unity3d.com/unity/whats-new/2018.3.6 consultado el 03/05/2019
[14] Página web: https://assetstore.unity.com/packages/tools/integration/steamvr-plugin-32647 consultado el 03/05/2019
[15] Página web: https://www.autodesk.com/products/maya/overview# consultado el 03/05/2019

- *Photoshop CS5 Extended*[16]. Es el programa que se utilizará para hacer las texturas de los objetos y personajes 3D mencionados anteriormente.

F 5.4 - Logo Photoshop

- *GitHub*[17]. Página web y aplicación de escritorio que permite tener el proyecto en internet y poder descargarlo y actualizarlo desde cualquier ordenador, disminuyendo el peligro de perder todo el progreso

F 5.5 - Logo GitHub

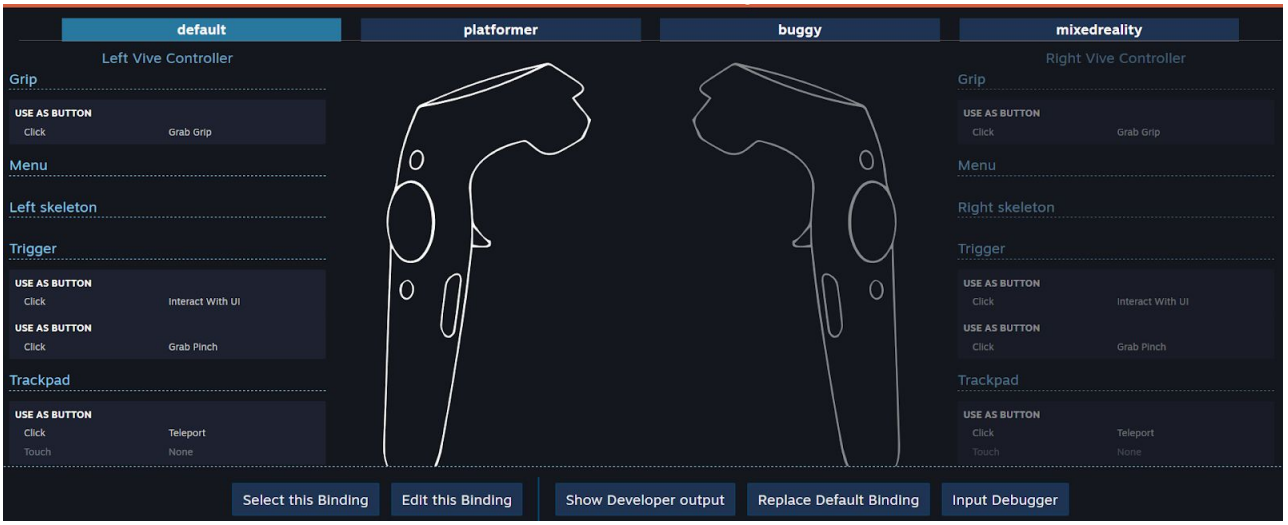
5.2 Desarrollo de mecánicas básicas y construcción del nivel

Para empezar el desarrollo en sí, lo primero que he hecho ha sido integrar el plugin de SteamVR en el proyecto de Unity. Para ello he tenido que configurar varias opciones. Para que cuando se ejecuta el juego y el jugador tiene puestas las gafas de realidad Virtual, pueda moverse por el mundo virtual dentro del área de juego pre-asignada sin ningún tipo de problema. Y también que se pueda ver el modelo 3D de los controladores dentro del juego (figura 5.6).

F 5.6 - Captura de los controladores en el juego

A continuación para seguir creando mecánicas básicas, he configurado los botones de los controladores con la interfaz gráfica de SteamVR para que cuando sean presionados, soltados o simplemente tocados hagan diferentes acciones enlazadas a ellos, visible en la siguiente figura 5.7.

[16] Página web: https://helpx.adobe.com/es/photoshop/get-started.html consultado el 03/05/2019
[17] Página web: https://github.com/ consultado el 03/05/2019



F 5.7 - Captura de la interfaz gráfica de SteamVR para enlazar acciones con botones

Las acciones que he implementado por ahora son la de coger objetos y la de teletransportarse, ya que son mecánicas básicas muy importantes en el juego que se está desarrollando. La acción de coger objetos se puede realizar ya bien presionando los botones laterales del controlador o el gatillo de debajo. Para hacer esto es necesario añadir un cubo invisible en la parte delantera de cada controlador y añadirle el siguiente script (Figuras 5.8, 5.9, 5.10):

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using Valve.VR;

public class ControllerGrabObject : MonoBehaviour
{
    public SteamVR_Input_Sources handType;
    public SteamVR_Behaviour_Pose controllerPose;
    public SteamVR_Action_Boolean grabAction;

    private GameObject collidingObject;
    private GameObject objectInHand;


    private void SetCollidingObject(Collider col)
    {
        if (collidingObject || !col.GetComponent<Rigidbody>())
        {
            return;
        }
        collidingObject = col.gameObject;
    }

    public void OnTriggerEnter(Collider other)
    {
        SetCollidingObject(other);
    }
```

F 5.8 - Captura número 1 del script ControllerGrabObject

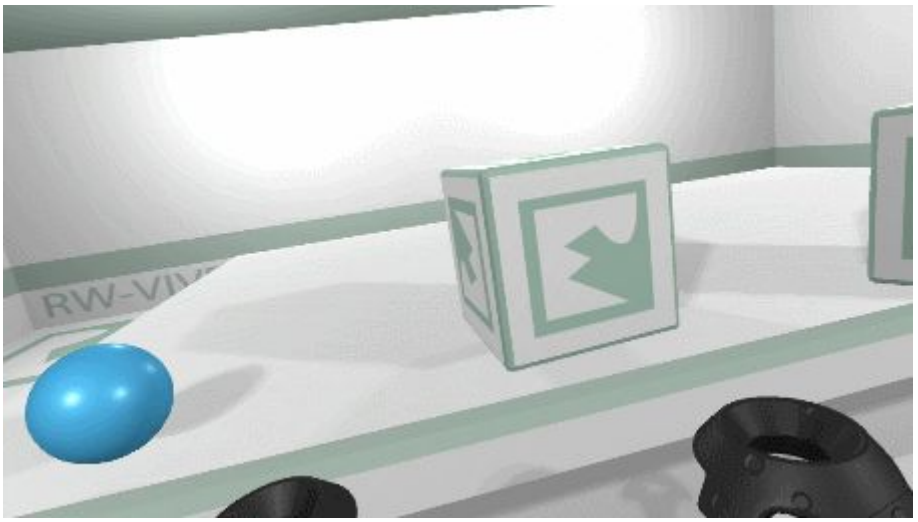
```
public void OnTriggerStay(Collider other)
{
    SetCollidingObject(other);
}

public void OnTriggerExit(Collider other)
{
    if (!collidingObject)
    {
        return;
    }
    collidingObject = null;
}

private void GrabObject()
{
    objectInHand = collidingObject;
    collidingObject = null;

    var joint = AddFixedJoint();
    joint.connectedBody = objectInHand.GetComponent<Rigidbody>();
}

private FixedJoint AddFixedJoint()
{
    FixedJoint fx = gameObject.AddComponent<FixedJoint>();
    fx.breakForce = 20000;
    fx.breakTorque = 20000;
    return fx;
}
```

F 5.9 - Captura número 2 del script ControllerGrabObject

```
private void ReleaseObject()
{
    if (GetComponent<FixedJoint>())
    {
        GetComponent<FixedJoint>().connectedBody = null;
        Destroy(GetComponent<FixedJoint>());

        objectInHand.GetComponent<Rigidbody>().velocity = controllerPose.GetVelocity();
        objectInHand.GetComponent<Rigidbody>().angularVelocity = controllerPose.GetAngularVelocity();

    }
    objectInHand = null;
}

// Update is called once per frame
void Update()
{
    if (grabAction.GetLastStateDown(handType))
    {
        if (collidingObject)
        {
            GrabObject();
        }
    }

    if (grabAction.GetLastStateUp(handType))
    {
        if (objectInHand)
        {
            ReleaseObject();
        }
    }
}
```

F 5.10 - Captura número 3 del script ControllerGrabObject

Una vez hecho eso y poniendo las opciones necesarias correctas en el inspector de unity, este es el resultado:



F 5.11 - Gif que muestra la mecánica de coger objetos

Durante esta primera parte he tenido muchos problemas con los controladores, ya que aun haciendo lo correcto, no se mostraban en el mundo virtual. Lo que ha solucionado esto ha sido crear el proyecto de unity directamente con el plugin SteamVR seleccionado como Asset Package en la ventana que sale al darle a crear un nuevo proyecto.

Para poder teletransportarse, primero he creado un láser a partir de un cubo el cual cambia de longitud dependiendo de dónde deba estar cada extremo. También es necesario agregar al controlador el siguiente script (figuras 5.12, 5.13, 5.14, 5.15) para que cuando se mantenga pulsado el touchpad del mando salga un láser en la dirección a la que se está apuntando y al mismo tiempo cuando se suelte el botón el jugador se teletransporte al lugar indicado:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using Valve.VR;

public class LaserPointer : MonoBehaviour
{
    public SteamVR_Input_Sources handType;
    public SteamVR_Behaviour_Pose controllerPose;
    public SteamVR_Action_Boolean teleportAction;

    public GameObject laserPrefab;
    private GameObject laser;
    private Transform laserTransform;
    private Vector3 hitPoint;

    public Transform cameraRigTransform;
    public GameObject teleportReticlePrefab;
    private GameObject reticle;
    private Transform teleportReticleTransform;
    public Transform headTransform;
    public Vector3 teleportReticleOffset;
    public LayerMask teleportMask;
    private bool shouldTeleport;
```

F 5.12 - Captura número 1 del script LasePointer

```
// Start is called before the first frame update
void Start()
{
    laser = Instantiate(laserPrefab);
    laserTransform = laser.transform;
    reticle = Instantiate(teleportReticlePrefab);
    teleportReticleTransform = reticle.transform;
}
```

F 5.13 - Captura número 2 del script LasePointer

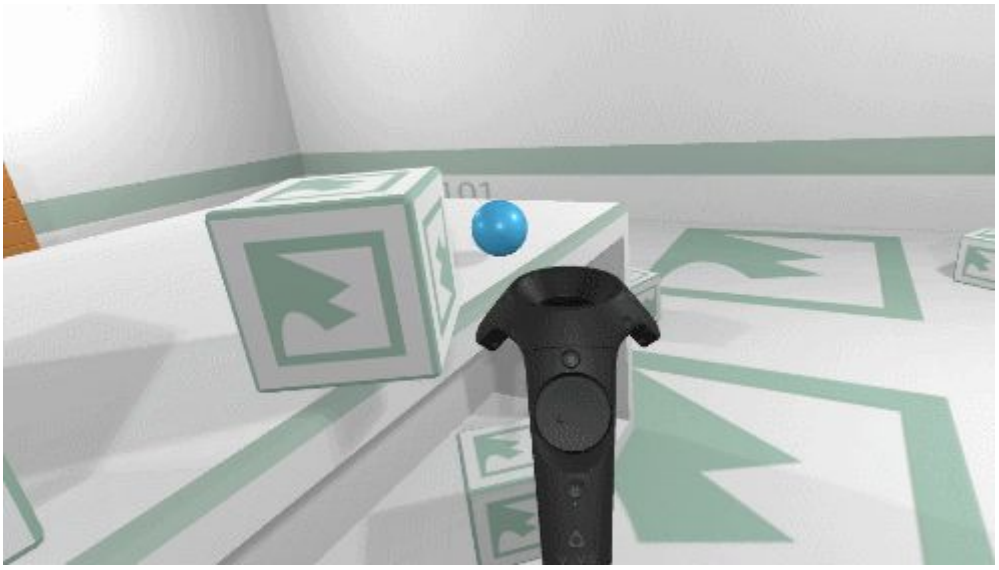
```
// Update is called once per frame
void Update()
{
    if (teleportAction.GetState(handType))
    {
        RaycastHit hit;

        if (Physics.Raycast(controllerPose.transform.position, transform.forward, out hit, 100, teleportMask))
        {
            hitPoint = hit.point;
            ShowLaser(hit);
            reticle.SetActive(true);
            teleportReticleTransform.position = hitPoint + teleportReticleOffset;
            shouldTeleport = true;
        }
    }
    else
    {
        laser.SetActive(false);
        reticle.SetActive(false);
    }

    if (teleportAction.GetStateUp(handType) && shouldTeleport)
    {
        Teleport();
    }
}
```

F 5.14 - Captura número 3 del script LasePointer

```
private void ShowLaser(RaycastHit hit)
{
    laser.SetActive(true);
    laserTransform.position = Vector3.Lerp(controllerPose.transform.position, hitPoint, .5f);
    laserTransform.LookAt(hitPoint);
    laserTransform.localScale = new Vector3(laserTransform.localScale.x, laserTransform.localScale.y, hit.distance);
}

private void Teleport()
{
    shouldTeleport = false;
    reticle.SetActive(false);
    Vector3 difference = cameraRigTransform.position - headTransform.position;
    difference.y = 0;
    cameraRigTransform.position = hitPoint + difference;
}
```

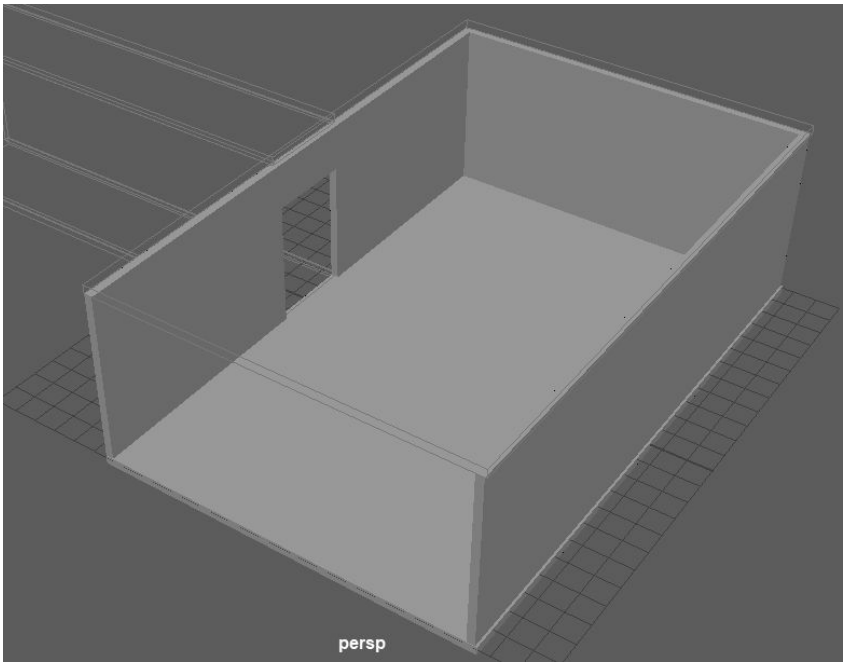
F 5.15 - Captura número 4 del script LasePointer

Después de añadir este script a los controladores y poner las opciones necesarias en el inspector de unity este es el resultado:

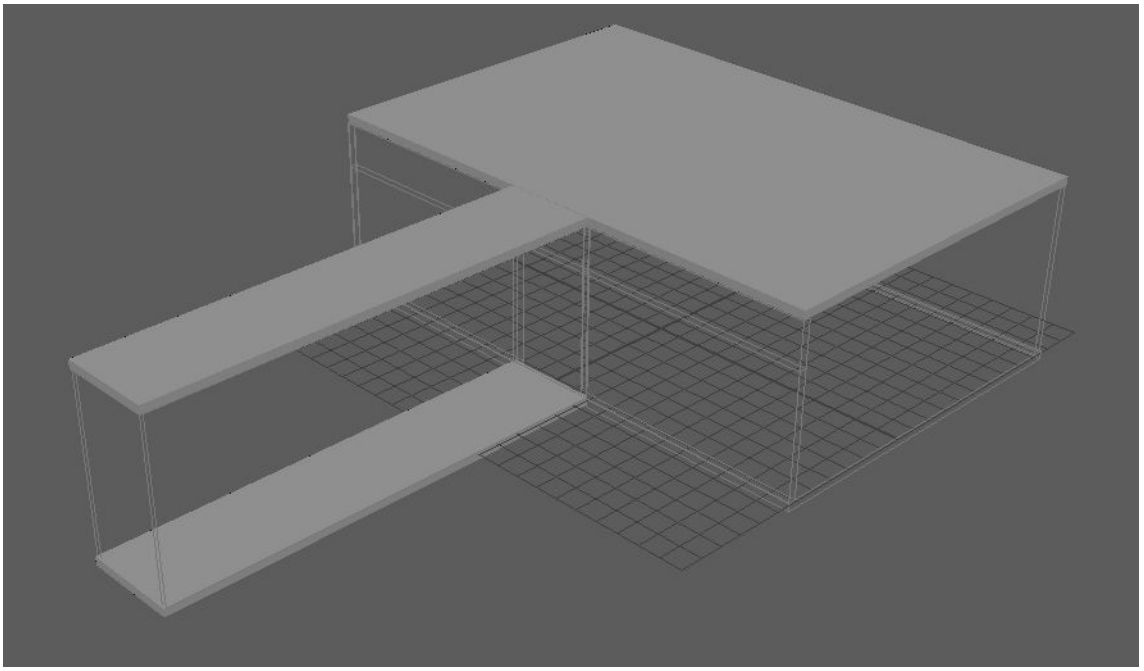


F 5.16 - Gif que muestra la mecánica de teletransportarse

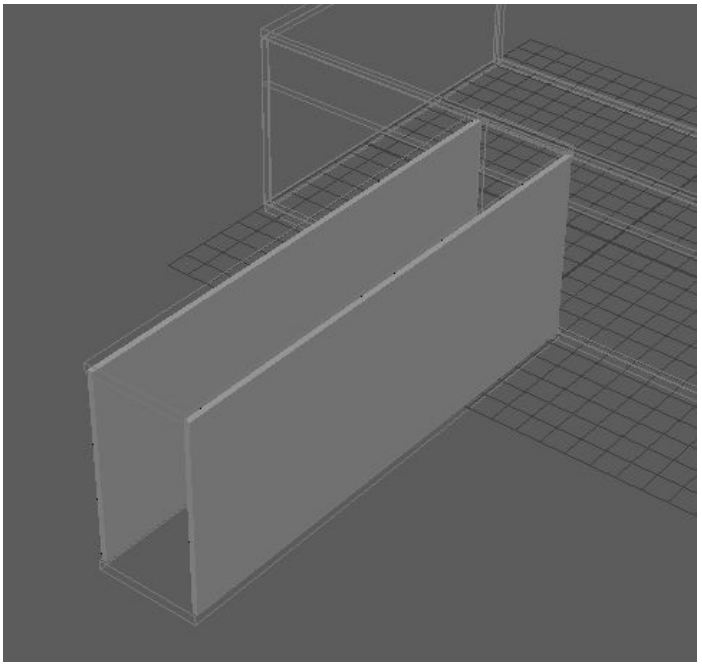
Para crear el escenario, he modelado las partes básicas estructurales. Tres paredes diferentes para crear las habitaciones en las que transcurrirá el juego, el suelo y techo para dichas habitaciones y finalmente las paredes, suelo y techo de los pasillos que conectarán las habitaciones.



F 5.17 - Captura de las tres paredes de las habitaciones y del suelo



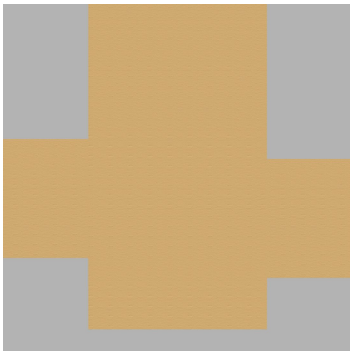
F 5.18 - Captura de los techos de las habitaciones y pasillo y del suelo del pasillo



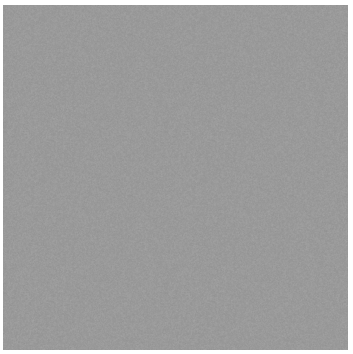
F 5.19 - Captura de las paredes del pasillo

A continuación he texturizado todos los modelos con unas texturas muy simples de madera para el suelo y de piedra para las paredes y los techos. Y finalmente lo he montado todo en mi escena de unity.

Aquí están las texturas:

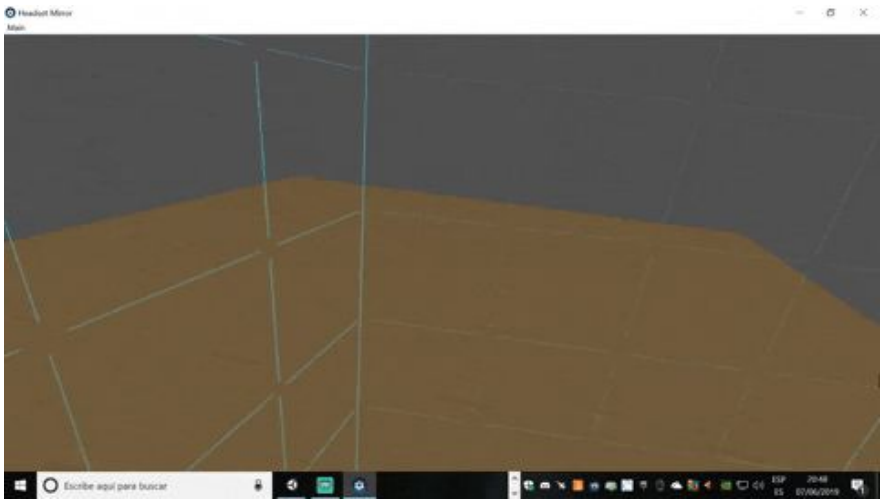


F 5.20 - Textura de madera de los suelos



F 5.21 - Textura de piedra para el resto de paredes

Y este es el resultado del nivel:



F 5.22 - Gif que muestra el escenario del juego

5.3 Desarrollo de mecánicas más complejas

En cuanto a los acertijos, ya que la idea en el juego es que cada vez que se inicie el juego en cada habitación aparezca un acertijo aleatorio, he creado un sistema en el que la textura que se le pone al objeto del acertijo sea una textura aleatoria de entre una lista. Estas texturas se asignan en el inspector de Unity y a la vez se guarda el ID de la textura en una propiedad del objeto. El código utilizado es relativamente simple:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class RandomRiddleTexture : MonoBehaviour
{
    public Texture2D[] textures;
    public int textureIDRoom = 0;

    void Start()
    {
        textureIDRoom = Random.Range(0, textures.Length);
        GetComponent<Renderer>().material.mainTexture = textures[textureIDRoom];
    }
```

F 5.23 - Código de randomizado de textura de acertijos

Así queda el acertijo en el juego, el texto cambia aleatoriamente al inicio de cada partida:

F 5.24 - Gif que muestra el acertijo en el juego

La respuesta de este acertijo es un objeto que habrá en la habitación. Cada objeto de la habitación tiene una propiedad con el ID de textura del acertijo al que corresponde. Por ejemplo, si toca aleatoriamente el acertijo en el que la respuesta es un diccionario (como en el ejemplo de la figura 5.24), el acertijo tendrá el ID 4 y el diccionario en sí que se encuentra en la escena tendrá una propiedad que será el número 4. En este momento del desarrollo los objetos aún son cubos blancos, con el único propósito de testear las funcionalidades. Para comprobar que los números de cada objeto coinciden con el acertijo (para saber si ellos son el resultado del acertijo) he utilizado el siguiente script, el cual debe estar en todos los objetos móviles de la escena:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class ObjectID : MonoBehaviour
{
    public int ID;
    public GameObject riddleContainer;
    public int riddleID;
    bool started = false;
    public bool matching = false;
    // Start is called before the first frame update
    void Start()
    {
        started = true;
    }

    // Update is called once per frame
    void Update()
    {
        if (started == true)
        {
            riddleID = riddleContainer.GetComponent<RandomRiddleTexture>().textureIDRoom1;
            started = false;
        }
        if (ID == riddleID)
        {
            matching = true;
        }
        else
            matching = false;
    }
}
```

F 5.25 - Captura del script ObjectID

El código de la figura 5.25 básicamente lo que hace es que en el caso de que el ID de la textura del acertijo y el ID del objeto coincidan el bool matching será verdadero. Este bool es el que más adelante utilizaré para ver si al intentar abrir la puerta es el objeto correcto. En caso de que sea verdadero, es el objeto correcto, y en el caso de que sea falso, es el objeto incorrecto.

Al abrir la puerta para que compruebe si el objeto que el jugador lleva en la mano es el adecuado, y así poder continuar si es correcto o perder una vida si no lo es, he añadido esta función al script ControllerGrabObject:

```
private GameObject objectInHandToCompareLeft;
private GameObject objectInHandToCompareRight;
```

F 5.26 - Captura de las propiedades añadidas al script ControllerGrabObject

```
private void CompareToOpen()
{
    if (leftHand.GetComponent<ControllerGrabObject>().objectInHand != null && rightHand.GetComponent<ControllerGrabObject>().objectInHand != null)
    {
        objectInHandToCompareLeft = leftHand.GetComponent<ControllerGrabObject>().objectInHand;
        objectInHandToCompareRight = rightHand.GetComponent<ControllerGrabObject>().objectInHand;

        if (objectInHandToCompareLeft.tag == "DoorHandle")
        {
            print("door is in left hand");
            if (objectInHandToCompareRight.GetComponent<ObjectID>().matching == true)
            {
                print("we are done here");
                //open door function
            }
            else
            {
                print("Wrong object");
                //lose a life
            }
        }
        if (objectInHandToCompareRight.tag == "DoorHandle")
        {
            print("door is in right hand");
            if (objectInHandToCompareLeft.GetComponent<ObjectID>().matching == true)
            {
                print("we are done here");
                //open door function
            }
            else
            {
                print("Wrong object");
                //los a life
            }
        }
    }
}
```

F 5.27 - Captura de la función CompareToOpen del script ControllerGrabObject

```
void Update()
{
    if (grabAction.GetLastStateDown(handType))
    {
        if (collidingObject)
        {
            GrabObject();
            CompareToOpen();
        }
    }
```

F 5.28 - Captura de como queda parte del Update del script ControllerGrabObject

En la figura 5.27 la función comprueba que haya un objeto en cada mano y en el caso de que uno de ellos sea el pomo de la puerta, procede a mirar si es el objeto de la otra mano es el correcto o no. En este punto he procedido a añadir una puerta a la escena que me he descargado de la Asset Store de Unity que se llama *Tim's Horror Assets - The Bloody Door*[18]. Aprovechando que ya tenía la animación hecha y que le daba muy buena ambientación al juego lo único que he hecho es añadirle los colliders necesarios a la puerta y al pomo, y ponerle al pomo el Tag de "DoorHandle" para que el código de la figura 5.27 reconozca el objeto como pomo de la puerta.

F 5.29 - Captura de la puerta con los colliders en el editor de Unity

Al pomo de la puerta le he añadido un script (figura 5.31) para que se ejecute en el caso de que el objeto sea el correcto. Este script comprueba que el bool de abrir la siguiente habitación sea verdadero y lo que hace es llamar a la animación Open de la puerta, que abre la puerta, y hacer que el jugador se pueda teletransportar al siguiente pasillo y habitación. Para hacer que los suelos sean teleportables lo único que hay que hacer es cambiar la layer del objeto de "NotTeleport" (que tiene asignado el número 10 de layer) a "CanTeleport" (que tiene asignado el número 9 de layer) como se muestra en la figura 5.31. Para que se ejecute este script he añadido el siguiente código a la función de la figura 5.27:

```
if (objectInHandToCompareLeft.tag == "DoorHandle")
{
    print("door is in left hand");
    if (objectInHandToCompareRight.GetComponent<ObjectID>().matching == true)
    {
        print("we are done here");
        objectInHandToCompareLeft.GetComponent<OpenNextRoom>().goToNextRoom = true;
    }
```

F 5.30 - Captura de la línea de código que pone en verdadero el bool necesario para la ejecución de abrir la puerta

[18] Puerta utilizada en el juego: https://assetstore.unity.com/packages/3d/props/interior/tim-s-horror-assets-the-bloody-door-70847 consultado el 08/06/2019

En cuanto este bool es verdadero se ejecuta el código del script del pomo de la puerta:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class OpenNextRoom : MonoBehaviour
{

    public bool goToNextRoom = false;
    public GameObject nextCorridorFloor;
    public GameObject nextRoomFloor;

    // Update is called once per frame
    void Update()
    {
        if(goToNextRoom == true)
        {
            transform.parent.transform.parent.GetComponent<Animator>().SetTrigger("open");
            nextCorridorFloor.layer = 9;
            nextRoomFloor.layer = 9;
            goToNextRoom = false;
        }
    }
}
```

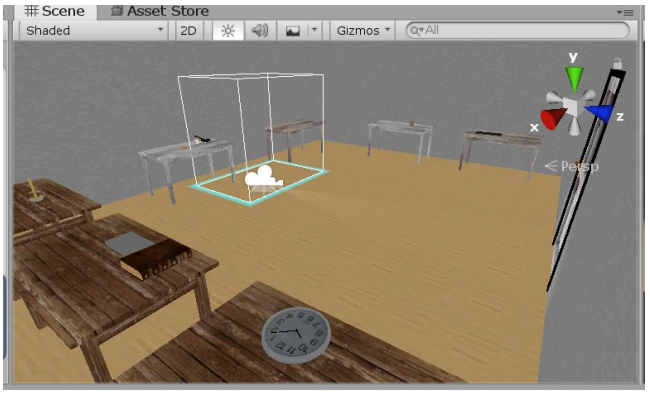
F 5.31 - Captura del script OpenNextRoom

5.4 Montado del nivel

Para poder comprobar si todo esto funcionaba, he decidido empezar a montar la primera habitación con los objetos que son totalmente necesarios. La mayoría de los objetos han sido adquiridos en las páginas web Free3D[19] y Turbosquid[20]. Estos objetos son:

- El objeto que contiene los acertijos aleatorios, sin textura, ya que la textura se asigna con el script de la figura 5.23.
- Mesas con sus respectivos colliders para poder dejar los objetos a una altura cómoda para que el jugador los pueda ver e interactuar con ellos.
- Los objetos que son las respuestas de los acertijos de esa habitación:
    - Un reloj de pared, un diccionario, una vela, un zapato, una moneda, un huevo y un teclado.

Este es el resultado de la primera habitación:



F 5.32 - Captura de la primera habitación en el editor de Unity

[19] Página web: https://free3d.com/ consultado el 09/06/2019
[20] Página web: https://www.turbosquid.com/ consultado el 09/06/2019

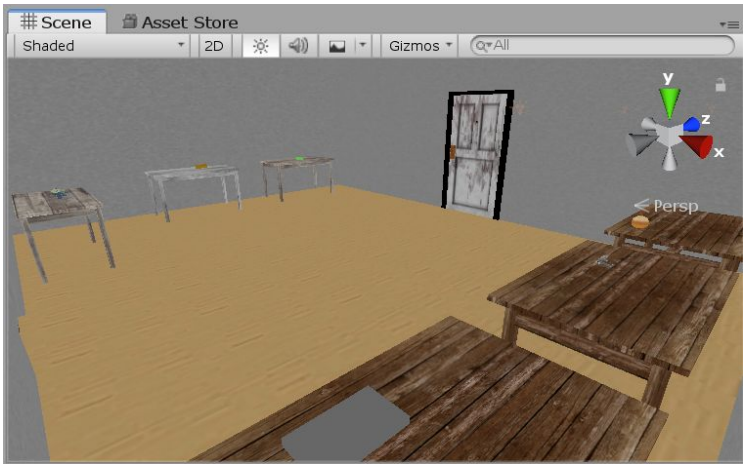
Estoy teniendo problemas con el mal texturizado de algunos objetos, ya que al tener que dedicarme a la separación de las UVs y el arreglo de las texturas, supone una pérdida de tiempo importante. Aunque, por otro lado, haya ahorrado tiempo al no tener que crearlos de cero.

Después de unas cuantas rondas de testeo, al ver que todo funcionaba con normalidad (los acertijos son aleatorios cada vez que se inicia el juego, si llevo el objeto incorrecto a la puerta no hace nada y si llevo el objeto correcto a la puerta, esta se abre y la siguiente habitación se vuelve teleportable) he decidido proceder a montar la segunda habitación de la misma manera que he montado la primera.

En esta habitación he puesto el objeto que contiene los acertijos aleatorios, las mesas que he utilizado en la primera habitación y los siguientes objetos:

- Una pastilla de jabón, un guante, una pistola, una cebolla y una esponja.

Este es el resultado de la segunda habitación:



F 5.33 - Captura de la segunda habitación en el editor de Unity

Testeando esta habitación, se me ha ocurrido intentar abrir la segunda puerta con el mismo objeto con el que he abierto la primera puerta. El resultado ha sido que la segunda puerta se ha abierto, cosa que no debería pasar. Para arreglar esto he añadido la siguiente línea al código que se muestra en la figura 5.30 para que una vez se haya usado el objeto correcto, deje de ser un objeto correcto:

```
if (objectInHandToCompareLeft.tag == "DoorHandle")
{
    print("door is in left hand");
    if (objectInHandToCompareRight.GetComponent<ObjectID>().matching == true)
    {
        print("we are done here");
        objectInHandToCompareRight.GetComponent<ObjectID>().matching = false;
        objectInHandToCompareLeft.GetComponent<OpenNextRoom>().goToNextRoom = true;
    }
```

F 5.34 - Captura de la línea añadida a la función CompareToOpen

Para que este bool no se vuelva a poner en verdadero en el siguiente frame, he tenido que modificar el Update del script ObjectID (figura 5.25) de la siguiente manera:

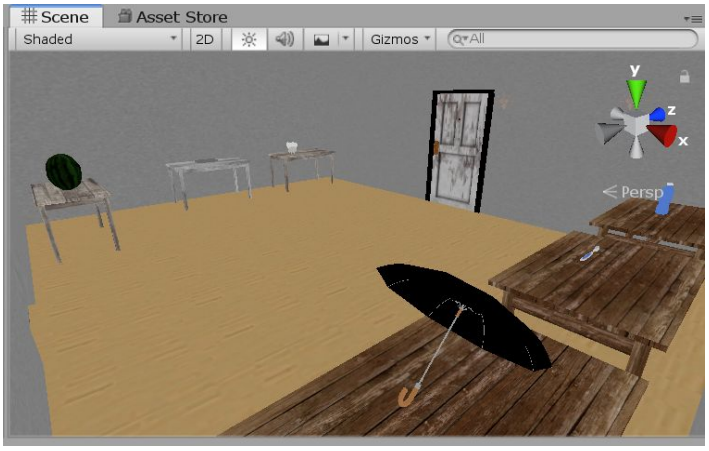
```
void Update()
{
    if (started == true)
    {
        riddleID = riddleContainer.GetComponent<RandomRiddleTexture>().textureIDRoom1;
        if (ID == riddleID)
        {
            matching = true;
        }
        started = false;
    }
}
```

F 5.35 - Captura del Update del script ObjectID

Al ver que esta modificación ha funcionado a la perfección, he procedido a montar la tercera habitación de la misma manera que he montado las dos habitaciones anteriores y la puerta que me quedaba en la tercera habitación. Los objetos que he añadido en esta habitación, a parte del objeto de los acertijos y las mesas, son los siguientes:

- Un diente, un cepillo de dientes, un paraguas, una botella de agua y una sandía.

Y este es el resultado de la tercera habitación:



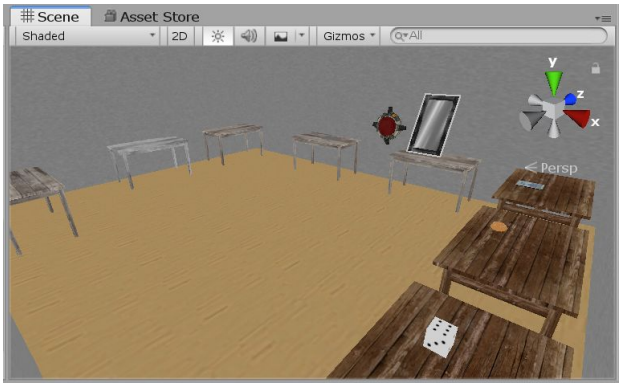
F 5.36 - Captura de la tercera habitación en el editor de Unity

Y finalmente solo queda montar la última habitación en la que los objetos son:

- Un donut, un dado, un espejo y un sello.

En esta habitación he colocado un botón que será el último objeto que comprobará si la elección del jugador es correcta, mostrando así si ha logrado superar el juego. La idea es que en caso de que gane, se le teletransporte a una habitación en la que pone que ha ganado, y lo mismo en el caso de que pierda (cuando cree un sistema de vidas), que se le teletransporte a una habitación en la que pone que ha perdido.

Resultado de la última habitación:



F 5.37 - Captura de la cuarta habitación en el editor de Unity

5.5 Desarrollo de sistemas de victoria y derrota

Para crear un sistema de vidas he creado un objeto que contiene el script que lleva la cuenta de las vidas, que en el caso que éstas lleguen a 0 el jugador será eliminado al momento y trasladado a una habitación donde pone Game Over en la pared. Cada vez que el jugador intente abrir una puerta o pulsar el botón del final con el objeto equivocado se ejecutará el siguiente código que he añadido a la función CompareToOpen de la figura 5.27, tanto para la mano derecha, como la izquierda:

```
if (objectInHandToCompareLeft.tag == "DoorHandle")
{
    print("door is in left hand");
    if (objectInHandToCompareRight.GetComponent<ObjectID>().matching == true)
    {
        print("we are done here");
        objectInHandToCompareRight.GetComponent<ObjectID>().matching = false;
        objectInHandToCompareLeft.GetComponent<OpenNextRoom>().goToNextRoom = true;
    }
    else
    {
        currentLives = lives.GetComponent<LivesCounter>().lives;
        currentLives = currentLives - 1;
        lives.GetComponent<LivesCounter>().lives = currentLives;
    }
}
```

F 5.38 - Captura de la parte de la función CompareToOpen en la que se restan las vidas

En cuanto las vidas lleguen a 0 entrará en acción el script LivesCounter el cual controla lo que le pasa al jugador cuando pierde:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using Valve.VR;

public class LivesCounter : MonoBehaviour
{
    public Transform cameraRigTransform;
    public Transform headTransform;
    public GameObject gameObjectToTeleport;

    public GameObject[] allFloors;

    public GameObject finalFloor;
    public bool teleport = false;
    public int lives;
```

F 5.39 - Captura de las propiedades del script LivesCounter

```
void Update()
{
    if(lives <= 0 && teleport == false)
    {
        teleport = true;
        Vector3 difference = cameraRigTransform.position - headTransform.position;
        difference.y = 0;
        cameraRigTransform.position = gameObjectToTeleport.transform.position + difference;
        RenderSettings.fog = false;
        for (int i = allFloors.Length; i > 0; i--)
        {
            allFloors[i-1].layer = 10;
        }
        finalFloor.layer = 9;
    }
}
```

F 5.40 - Captura del Update que controla lo que pasa cuando el jugador llega a 0 vidas

Con esta función no he tenido ningún problema, ya que es algo muy parecido a lo que se hace cuando el jugador se teletransporta utilizando el láser.

En el caso de que el jugador tenga el objeto adecuado mientras pulsa el botón y tenga que ser transportado a la habitación que muestra la victoria, he añadido el siguiente código a la función CompareToOpen de la figura 5.27, para la mano derecha y la izquierda, en este caso solo mostraré el de la mano izquierda:

```
if (objectInHandToCompareLeft.tag == "EndButton")
{
    print("button is in left hand");
    if (objectInHandToCompareRight.GetComponent<ObjectID>().matching == true && teleport == false)
    {
        print("we are done here");
        objectInHandToCompareRight.GetComponent<ObjectID>().matching = false;
        //Teleport to final room
        teleport = true;
        Vector3 difference = cameraRigTransform.position - headTransform.position;
        difference.y = 0;
        cameraRigTransform.position = gameObjectToTeleport.transform.position + difference;
        RenderSettings.fog = false;
        for (int i = allFloors.Length; i > 0; i--)
        {
            allFloors[i - 1].layer = 10;
        }
        finalFloor.layer = 9;
    }
    else
    {
        currentLives = lives.GetComponent<LivesCounter>().lives;
        currentLives = currentLives - 1;
        lives.GetComponent<LivesCounter>().lives = currentLives;
    }
}
```

F 5.41 - Captura de la parte de código que controla lo que pasa cuando pulsas el botón de la cuarta habitación

5.6 Ambientación del prototipo

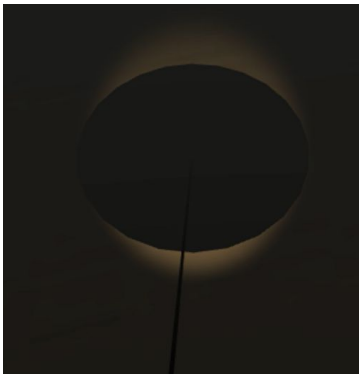
En este momento del desarrollo la parte de puzzle ya está completa, ahora me centraré en hacer que el jugador se sienta en tensión mientras intenta solucionar los acertijos y acercarme más al género de Terror.

Lo primero que he hecho es poner la iluminación de la escena muy oscura para que el jugador vea mucho menos y añadir niebla para darle un toque más tenebroso y hacer que el jugador se sienta un poco incómodo, ya que ese es el objetivo del juego. Para que el jugador pueda ver

algo con más claridad le he añadido una luz al mando izquierdo, simulando una linterna. Este es el resultado in-game:

F 5.42 - Captura del juego en la que se ve la oscuridad y niebla

En este punto, al teletransportarse, el láser que marca la posición a la que el jugador se quiere teletransportar no encaja para nada con el estilo del juego, por lo que he decidido ponerlo todo totalmente negro, el láser y la marca que aparece en el suelo. Este es el resultado:



F 5.43 - Captura del juego en la que se ve el láser y la marca de color negro

Para ambientar mejor las habitaciones también he añadido antorchas en las habitaciones, cada una con un generador de partículas en el que se simula fuego, y con unas luces tenues.

F 5.44 - Captura de una antorcha del juego

Finalmente he añadido una música ambiente en loop que mantiene al jugador en tensión durante todo el juego y un efecto de sonido que asuste al jugador en el momento que falla al intentar abrir una puerta o pulsar el botón final con el objeto incorrecto.

5.7 Optimización

En cuanto a la optimización, no le he prestado mucha atención en este proyecto. Simplemente he hecho que el far plane de la cámara esté mucho más cerca, ya que habiendo niebla, la visión del jugador es muy limitada. Por otro lado, al haber tan pocos objetos, ha sido innecesario el uso de occlusion culling, que disminuye el consumo de GPU dejando de renderizar los objetos ocultados por otros.

5.8 Testeo

He realizado testeos con dos personas, Ian Trueba Y Salvi Carreras, que acostumbran a jugar todo tipo de videojuegos, para conocer su opinión sobre el prototipo y saber que cambiarían en un futuro.

Ambos han coincidido en que hay muy pocos objetos en la escena entre los que elegir cuál es la respuesta correcta. También han coincidido en que se sentían completamente inmersos en el mundo virtual mientras jugaban, aunque Salvi de una manera más intensa que Ian.

Ian no se ha sentido para nada en tensión con la oscuridad ni el sonido, en cambio Salvi si que se ha sentido en tensión.

A Salvi no le quedaba claro si al abrir la puerta solo debía pulsar el gatillo al acercarse al pomo o si necesitaba girar la mano también. Después de unos cuantos intentos ha podido observar que solo debía apretar el gatillo y la puerta se abría automáticamente.

En cuanto a los cambios a hacer en el futuro, Salvi ha dicho que estaría bien añadir distintos posibles caminos y que los pasillos no sean una línea recta, sino que haga giros de ángulos rectos para que no sepas lo que te puedes esperar detrás de la esquina. Ian ha dicho que le gustaría ver muchos más objetos en la escena, y que a ser posible que los enigmas estén un poco escondidos para que el jugador tenga que buscarlos, por ejemplo debajo de una mesa.

Los dos han dicho que han disfrutado del prototipo, lo que no se si se debe a que tengan poca experiencia con la Realidad Virtual y que lo hayan disfrutado simplemente porque les ha gustado sentir que estaban en lugar completamente diferente.

6. Conclusiones y trabajos futuros

Este trabajo me ha ayudado a aprender todos los entresijos necesarios para desarrollar un juego en Realidad Virtual. Desde que hay que controlar la manera exacta en la que se mueve el jugador para que no se maree, hasta el frame rate que debe ejecutarse el juego, por las mismas razones... Pero sobre todo a saber utilizar el plugin SteamVR, gracias al cual he podido trabajar de una manera más sencilla y eficiente para conseguir el resultado que deseaba. Utilizando esta herramienta he podido observar que los desarrolladores del SteamVR han dedicado mucho tiempo para que los desarrolladores de videojuegos tengan todas las funcionalidades que necesiten para hacer cualquier tipo de juego que se propongan en Realidad Virtual.

Durante el desarrollo me he encontrado con varios problemas, siendo el más importante la inversión inesperada de tiempo en producir diferentes funcionalidades, debido a la aparición de nuevos problemas tras la solución de otros.Por otro lado, he notado mucho la falta de un artista que me suministrara con los modelos 3D ya texturizados. He tenido que buscar fuentes por internet, descargar modelos, la mayoría mal texturizados y arreglarlos. Teniendo en cuenta que tengo un perfil de diseñador/programador este ha sido un problema difícil de superar y el motivo por el que no hay tantos objetos que deseaba.

Otro de los grandes problemas deriva de la relación de mi trabajo con el de otro compañero. Al no poder llegar al resultado esperado por su parte, he notado una falta de feedback muy necesario para hacer el juego más de Terror que de Puzzle.

Revisemos los objetivos específicos antes de iniciar el estudio sobre la Realidad Virtual y el desarrollo del prototipo del juego:

- Que sea totalmente inmersivo
    - Este objetivo se ha cumplido en su totalidad, gracias al sonido ambiente y el aspecto inmersivo que viene con hacer un juego en Realidad Virtual.
- Que sea divertido y rejugable
    - Todos los jugadores que han podido probarlo han acabado satisfechos con la experiencia, y gracias a hacer que los enigmas sean aleatorios cada vez que se ejecuta el juego se ha conseguido que sea rejugable.
- Que cree un sentimiento de ansiedad al intentar superar los obstáculos presentados por el juego
    - Se apunta como un punto mejorable tras conseguir los datos necesarios de los que carezco (mencionado anteriormente).
- Deberán haber 5 habitaciones diferentes en la que tienes que resolver un enigma y recoger un objeto de la habitación (el cual debe ser la respuesta al enigma realizado) para pasar a la siguiente.
    - Este punto se da por cumplido, aún habiendo creado 4 habitaciones en lugar de 5, ya que la mecánica es la misma.
- 10 enigmas diferentes en cada habitación que se eligen de manera aleatoria al empezar a jugar.
    - Este objetivo se ha quedado a medias, ya que solo he llegado a tener 20 enigmas aleatorios sumando entre las 4 habitaciones. Este punto habrá que trabajarlo más en el futuro.

Gracias a los testeos he visto qué funcionalidades han conseguido un mejor y peor resultado. Las conclusiones que saco al respecto son las siguientes:

- Me he quedado corto en cuanto a los objetos interactuables de las habitaciones.
- El juego no pone lo suficiente en tensión al jugador como para que se considere que el jugador está pasando miedo.
- Los jugadores prefieren utilizar la función de teletransporte a puntos muy cercanos en vez de andar hacia ellos aunque se encuentren dentro del área de juego. Hay que recalcar que los testers del prototipo se han encontrado en una área de juego reducida y que si las condiciones espaciales fueran óptimas se pondría en duda este punto.
- La luz simulando la linterna en el controlador izquierdo ha sido un gran acierto, ya que al estar tan oscuro los jugadores la necesitaban para poder leer cómodamente los acertijos.

En cuanto a trabajos futuros queda acabar el juego por completo, tomar todo el feedback recibido y aplicarlo para conseguir un mejor resultado.

Para acabar totalmente el juego, será necesario hacer lo siguiente:

- Añadir al antagonista llamado Evil Sage, el cual se dedique a asustar al jugador.
- Añadir toda la narrativa que envuelve a Evil Sage.
- Hacer una habitación en la que el jugador se ponga unas gafas de Realidad Virtual que le cambien por completo los alrededores, tal como se explica en el documento de diseño.
- Cambiar el diseño del nivel para conseguir más tensión en los jugadores.
- Añadir elementos que asusten al jugador, por ejemplo screamers.
- Producir una cantidad ingente de objetos para que cueste mucho más encontrar las respuestas entre tanto objeto.
- Añadir muchos acertijos más para que el juego sea aún más aleatorio y rejugable.
- Hacer más uso de audios para controlar las emociones del jugador.
- Optimizar el juego todo lo posible para que el rendimiento sea excepcional y que los jugadores no tengan problemas de mareos.

7. Bibliografía

- ¿Qué es la Realidad Virtual?
    - http://mundo-virtual.com/que-es-la-realidad-virtual/ consultado el 05/03/2019
- Los movimientos oculares
    - http://www.ub.edu/pa1/node/movimientos_oculares consultado el 06/03/2019
- The best VR horror games
    - https://www.pcgamer.com/the-best-vr-horror-games/ consultado el 08/03/2019
- HTC Vive Tutorial for Unity
    - https://www.raywenderlich.com/792-htc-vive-tutorial-for-unity consultado el 09/03/2019
- Basic Unity Tutorial for Steam VR & Vive (Setting up HMD and controllers)
    - https://www.youtube.com/watch?v=LZTctk19sx8 consultado el 09/03/2019
- El proceso productivo del videojuego: fases de producción
    - https://revistas.ucm.es/index.php/HICS/article/download/45178/42539 consultado el 14/03/2019
- What is it riddles
    - https://www.riddles.com/what-is-it-riddles?page=1 consultado el 29/04/2019
- Free3D models
    - https://free3d.com/3d-models/ consultado el 02/06/2019
- Turbosquid (más modelos 3D)
    - https://www.turbosquid.com/ consultado el 01/06/2019
- Unity Scripting API
    - https://docs.unity3d.com/ScriptReference/ consultado el 03/06/2019

8. Anexos

The Evil Sage

GAME DESIGN DOCUMENT

Asier Iglesias Gavarró

Revision: 0.2.0

Overview of the Game Design Document

Theme / Setting / Genre

- VR Terror

Core Gameplay Mechanics Brief

- Solve riddles
- Grab objects
- Teleport through rooms
- Walk in real life

Targeted platforms

- HTC Vive/PC

Monetization model

- Premium

Project Description:

This game is a single player VR terror game, in which you need to solve riddles in order to advance through the game and free yourself from the horrific VR experience you have been dragged into without your consent.

There are a few rooms full of objects, where in each one of them you need to find the riddle and solve it, the answer being one of the objects that can be found in the room. After finding this object you need to grab it and take it to the door in order to get to the next room. Be careful and get the correct object, or you'll be punished by the Evil Sage.

What sets this project apart?

- There's no other game with the same mechanics and gameplay.

Core Gameplay Mechanics (Detailed)

- Walk in real life

- Walk around inside your personally designated area

For this to work you need to have the SteamVR installed and have the area of play already configured.

- Grab objects

- Grab an object in order to move it to the objective

To do this use the trigger of the controllers while touching the object.

- Teleport through the room

- Teleport where you point to with the controller

To do this keep the touchpad of any controller pressed and a laser pointing at a place will appear. Just release the touchpad when the laser is pointing at the place you want to teleport.

- Solve riddles

- In each room you need to find the riddle and solve it.

By solving the riddle you'll know which object you need to grab and take to the next room.

Riddles

- What has a head, a tail, and has no legs? A coin.
- When does Christmas come before Thanksgiving? In the dictionary.
- My life can be measured in hours, I serve by being devoured. Thin, I am quick. Fat, I am slow. Wind is my foe. What am I? I am a candle.
- What is black when you buy it, red when you use it, and gray when you throw it away? Charcoal.
- What is more useful when it is broken? An egg.
- What has six faces, But does not wear makeup. It also has twenty-one eyes, But cannot see? A dice.
- They have not flesh, nor feathers, nor scales, nor bone. Yet they have fingers and thumbs of their own. What are they? Gloves.
- What goes up when the rain comes down? An umbrella.
- I am full of holes but I can still hold water. What am I? A sponge.

- Always in you, Sometimes on you; If I surround you, I can kill you. What am I? Water.
- I make two people out of one. What am I? A mirror.
- I have two hands, but I can not scratch myself. What am I? A clock.
- There was a green house. Inside the green house there was a white house Inside the white house there was a red house. Inside the red house there were lots of babies. What am I? A watermelon.
- (What's at least 6 inches long, goes in your mouth, and is more fun if it vibrates? A toothbrush.)
- What travels around the world but stays in one spot? A stamp.
- I'm not clothes but I cover your body; The more I'm used, the thinner I grow. What am I? Bar of Soap.
- I'm white, and used for cutting and grinding. When I'm damaged, humans usually remove me or fill me. For most animals I am a useful tool. What am I? A tooth.
- (A time when they are green, a time when they're brown, but both of these times, cause me to frown. But just in between, for a very short while, They're perfect and yellow and cause me to smile! What am I talking about here? Bananas.)
- You use a knife to slice my head and weep beside me when I am dead. What am I? An onion.
- Hands she has but does not hold, teeth she has but does not bite, feet she has but they are cold, eyes she has but without sight. Who is she? A doll.
- What has no beginning, end, or middle? A doughnut.
- You use me for my name, I'm not a breeze to tame, I'm fastest when I'm full and when it's cheap it sounds the same. You'll spot a nest near me (although I'm not a tree), the sea is fore, the sea is aft, it's all around, you see. What am I? A sail.
- What is it that is full all-day and empty at night? Shoes.
- Slam slam slam all day long slam slam slam some fast, some slow something solid. flat and sturdy its friend lights up the night and is sensitive to the eye slam slam slam A through Z 1,2,3 black as night. What am I? A Computer keyboard.
- What stays where it is when it goes off? A Gun.

- I exist, but have no material form. I am made of numbers, but appear square. I build upon other of my own, but do not topple down. I sit still and do nothing, but I'm still useful. What am I? Minecraft Block.

Story and Gameplay

Story

You appear trapped inside a room with no exits and there's a weird box on a table with a sign asking you to put on the VR glasses that are inside it. As soon as you enter the Virtual reality world you are playing on the Evil Sages hands, and for it to end you need to play a game of solving riddles through different rooms.
The Evil Sage is a horrendous being of another dimension, who feeds on the fear of the people who play his game.

Gameplay

Being at that room, the first to do to start playing the game is to put on the virtual glasses (the ones inside the game) by grabbing them with the controller pressing the trigger. As soon you put on the glasses the environment will change and you will appear in another completely different room.

Inside the that room you will find a lot of objects around the room and by moving around it, you need to find the frame where the riddle is. After solving it (the answer will be one of the objects in the room), you need to grab that object to the door that has been closed the entire time. In order to open the door use the trigger of the controller that's free (in the other hand you need to keep the other object grabbed). After this you will need to walk/teleport through a corridor to the next room, and the story will repeat, find the riddle, solve it and get the correct object, then open the door and to the next room. After doing the same a few times, you will get to the room where the Evil Sage is resting.

You only have two tries to guess the right object, if you get the wrong object you will lose one of the tries and the game will make you to be more anxious. If you get the incorrect object a second time the Sage will absorb all your life energy and you will die.

Even if you get the wrong object the first time, the door will open and you will be able to continue, but everything will turn darker and the music more intense.

Assets Needed

- **2D**
    - Textures
        - Environment Textures
        - Texture for the 3D character
        - Texture for all the environmental art
- **3D**
    - Characters List
        - The Evil Sage.
    - Environmental Art Lists
        - The actual environment (the house where all happens).
    - Key objects
        - Different tables and shelves (as much as possible).
        - VR headset.
        - A frame to put the riddles.
        - A coin.
        - A dictionary.
        - A candle.
        - Charcoal.
        - An egg.
        - A dice.
        - Gloves.
        - An umbrella.
        - A sponge.
        - A Water bottle.

- A mirror.
- A clock.
- A watermelon.
- A toothbrush.
- A stamp.
- A bar of soap.
- A tooth.
- Bananas.
- An onion.
- A doll.
- A doughnut.
- A sail.
- A Shoe.
- A Computer keyboard.
- A Gun.
- A Minecraft Block.

Extra Objects

-

- **Sound**

- Sound List (Ambient)
  - Inside
    - Phil Michalski - PARANOIA_Drone_12

- Sound List (Player)
  - Character Fail
    - Phil Michalsk - PARANOIA_IMPACT_11

- **Code**

- Character Scripts (Player Pawn/Player Controller)
  - Teleport script
  - Grab Object script
  - Open Door script

- Ambient Scripts (Runs in the background)

        - Random Riddle script
    - NPC Scripts
        - Evil Sage Events script.

- **Animation**

    - Environment Animations
        - Door opening animation
    - Character Animations
        - The Evil Sage
            - Walking animation
            - Scare the player animation
            - Fly to the camera animation